Arthur Ogawa

Arthur Ogawa is a long time member of the TUG board and well-known package developer.

[Interview completed 14 July 2008.]

Dave Walden, interviewer: Please tell me a bit about your personal history independent of TEX.

Arthur Ogawa, interviewee: I am a high-energy particle experimental physicist (Ph.D., UC Berkeley, 1978), now in my second career as a computer consultant in the field of electronic publishing.

I was born in Milwaukee, Wisconsin, in 1948, did my undergraduate work at the University of Wisconsin, Madison, and California Institute of Technology, Pasadena (BS, 1970).

I live with my wife Marian Goldeen and children Grace (20) and Evan (18) in Three Rivers, California, an unincorporated village in the foothills of the Sierra Nevada, at the entrance to Sequoia National Park.

I am active in my local community, working on local food sources, among other things. I ride my bikes (mountain and road) in the best bicycling area in the world.

DW: When and how did you first get involved with TEX?

AO: I had been working as a post-doc at the Stanford Linear Accelerator Center (SLAC) and had had experience with earlier computer-based systems for the creation of technical documents. Some time about 1980, I heard about a new system authored by my old math professor, who was then working on the Stanford University campus.

I had taken a course in algebra from Donald Knuth when I was a sophomore at Caltech, and I was now eager to try out TEX82, version 0.9999, which I found running under VMS on a Digital Equipment VAX 11/780. I ran a sample document through, and printed it out on the local printer, a Canon LBP-10: I was hooked!

Notwithstanding the remarkably obscure error message I encountered when I made one small (invalid) change to the source file on my second run. This was a humbling experience, indeed.

I obtained the manual (by going to the Computer Science office down on the Stanford campus), studied it thoroughly, and I was off and running. I became one of the ten institutional *TUGboat* subscribers at SLAC, got to know the other TEX users well, and started writing my own macro package — still available on CTAN, it is called Psizzl (`http://mirror.ctan.org/macros/psizzl`).

Over time I was asked to help people use TEX on their own documents, and I became

familiar with the experience of deciphering `\tracingall` output at 3am.

About 1985 or so, I helped Dr. Anthony Siegman and his TEX-based typesetter, Laura Friedman, complete his book on Lasers (University Science Books, see `http://www.tug.org/TUGboat/Articles/tb08-1/tb17complete.pdf`, page 8). It was my first venture into TEX consultancy. I eventually purchased a Macintosh Plus (1MB RAM, fitted out with a 20MB disc drive) and ran FTL Systems' MacTEX.

By 1987, I had formed my modest firm, TEX Consultants, in Palo Alto, California. After a call from Lynne A. Price (who herself had served on the TUG Board), I began working for Hewlett-Packard (HP) at their corporate offices. Their SGML-based publishing system HPtag was much like the XML-based publishing systems of today, in use by, for example, the American Physical Society. In the case of HPtag, documents were first coded in SGML, validated, and then translated into TEX for typesetting by the macro package I wrote. HP product manuals of all kinds were produced using HPtag.

By this means I supported myself and my growing family. I worked for any number of corporate clients, including Apple Computer, the NASA Ames Research Center, and others. I also typeset books for publishers, producing films suitable for use in offset printing.

I began collaborating with William E. Baxter, another TEX-based typesetter and programmer. He had attended UC Berkeley math department at the same time as my wife; upon graduation, he went into the business as SuperScript (`superscript.com`).

I was a pioneer of working in color printing with TEX. At the 1989 TUG conference at Stanford University, I demonstrated how I was using TEX to create color separations (through the vehicle of PostScript). I even showed how TEX could accommodate the need for trapping, something that few people in the audience understood or appreciated. Afterwards, Don Knuth asked me for my opinion on how well TEX worked with color.

With 20/20 hindsight, I now wish I had responded differently than I did: I believed at the time that color could be handled adequately by putting `\special` commands into the DVI output. Unfortunately, I was wrong. It became clear shortly thereafter that the color of the ink is in all ways like the font of the glyph: TEX should keep track of both attributes on the same footing: those familiar with TEX's internals know that each glyph "remembers" its font, and the color of the glyph's ink should be handled in the same way. (Rules should remember their ink color as well.)

The upshot of my conversation with Knuth was that he made major changes to TEX in 1989 (when TEX version 3.0 came into being) without addressing the issue of color. The problem remains with us today. For an example of the problem, create a document with a colored footnote that breaks over a page.

After working with many publishers and corporate clients in the Bay Area, the advent of the Internet enabled me to perform all my work out of my home office, and we relocated from the Bay Area to Three Rivers in 1993.

DW: Please describe what you are currently doing in the world of TEX.

AO: I am presently developing macro packages for the American Physical Society and the American Institute of Physics.

The APS authoring package REVTEX 4 was released in 2001; the release of REVTEX 4.1 is under development now (spring 2008) and will incorporate a module that covers the journals of the AIP.

DW: You wrote an article in *TUGboat* issue 22:3 (September 2001) about version 4.0 of REVTEX.

AO: REVTEX 4 is a LATEX 2_ε document class, a complete rewrite of REVTEX 3, itself a

LaTeX 2.09 style dating back to the late 1980s.

Work on REVTEX 4 was started by David Carlisle; I took over his pilot project. I added powerful low-level machinery based on the work of William Baxter (the `ltxutil` and `ltxgrid` packages), enabling two-column typesetting and other features. REVTEX 4 uses the natbib package of Patrick W. Daly, and its BIBTEX styles are generated using Daly's custom-bib package, both of which were extended for the sake of REVTEX 4. This work was done by David Carlisle, Patrick Daly and me, under contract to the APS with Mark Doyle as the contract officer. Dr. Baxter has graciously allowed his inventions to be used gratis.

The `ltxutil` package provides OOP capabilities along with more primitive things, notably an algebra that eliminates the explicit use of `\if` and `\fi` (making the code easier to debug).

The `ltxgrid` package completely redoes LaTeX's output routine and float placement algorithm, allowing among other things the placement of both single-column and full-page-width floats in the same multicolumn document.

REVTEX 4 tries hard to achieve compatibility among the many LaTeX supported packages; it even has code that allows David Carlisle's `longtable` package to work properly in a multicolumn layout, something that you cannot accomplish with LaTeX's own multicol package.

DW: When I look up REVTEX (`http://authors.aps.org/revtex4/`), it appears to be a fairly major package.

AO: REVTEX is a fourth-generation macro package. Earlier work I did entailed a nearly complete rewrite of LaTeX 2.09, predating LaTeX 2_ε. William Baxter and I worked cooperatively on that package, using it to create commercial typesetting systems. The NASA Ames Research Center's NASATEX is based on that work.

DW: Issue 15:3 (September 1994) of *TUGboat* shows that you gave a presentation on "Object-oriented programming, descriptive markup, and TEX".

AO: I gave that talk at the Santa Barbara TUG meeting (1994); William Baxter gave a companion talk describing his OOP extensions to TEX. TEX macros implementing OOP and the new output routine are now part of the `ltxutil` and `ltxgrid` packages.

DW: Issue 20:3 of *TUGboat* (September 1999) shows you participated in a panel on the Future of LaTeX and gave a talk on doing database publishing involving Java and using TEX as the typesetting back end.

AO: The Vancouver TUG meeting featured the "Future of LaTeX" panel, which I chaired. On the panel were the movers and shakers of LaTeX 2_ε and LaTeX 3. I challenged them by pointing out that they had taken on the task of bringing out a successor to Leslie Lamport's LaTeX, but despite the passage of 10 years, they had failed. Afterwards, friends told me "that was really bad." I thought it was exciting. We are now coming up to the 20-year anniversary of Team LaTeX's inheritance of the LaTeX development responsibilities. Perhaps we should have another such panel discussion.

Database publishing has always been a keen interest of mine. The talk I gave at that conference had to do with a system I created for Patrick Chan, the author of the Java Almanac. Dr. Chan would aim a Java program at Sun Microsystem's internal documentation for Java, turn the crank, and out would pop his book.

TEX's markup language is a terrible one for the author — if he is not a computer. On the other hand, a computer program can talk to TEX just fine, thank you. Don Knuth anticipated this use of TEX at the outset. It's a wonder that people are still coding TEX

documents by hand at this late date.

DW: Earlier in this interview you mentioned a SGML-based publishing system (HPtag). So clearly you have spent some time thinking about how TEX fits into the world of typesetting and publishing and computing more generally. Please tell me your current thinking about TEX and its current and future utility.

AO: TEX has two severe drawbacks: its front end, pertaining to document creation, and its interface to the macro writer, the programming interface.

As early adopter Dr. Anthony Siegman understood, descriptive markup should be the only TEX commands to be used in the document instance. Leslie Lamport also understood this well, as his remarks at the Santa Barbara TUG meeting made clear. But, as Theodor Holm Nelson explains in ("Embedded Markup Considered Harmful" (`http://www.xml.com/pub/a/w3j/s3.nelson.html`), it does not go without saying that embedded markup is itself correct. So, TEX should really have a front end, such as is supplied by Scientific Word (`http://www.mackichan.com/`). Note that in the case of that software, the internal representation of the document is a collection of objects, and the LATEX form is used only for document storage and exchange. Other representations are therefore possible. Meanwhile, the document maintainer (the "author") does not have to concern herself with markup at all.

The programming interface is how we macro writers create an abstract formatting engine. For example, this is what I do when I create a LATEX document class. However, dealing with the macabre TEX programming language is not the right way to get the job done. A cleaner interface is provided by, e.g., FrameMaker or QuarkXPress, where your interface is a panel with sliders, menus, and other modern UI tools. In fact, PCTEX is now offering such a system, and Scientific Word had one back in the early 1990s.

At the same time, progress has left TEX's markup paradigm behind. With the advent of Unicode, it is no longer necessary to use control sequence names for many glyphs, like `\alpha`. Unicode itself evolved alongside of TEX, and it is really too bad that TEX was not Unicode-aware in the first place.

People are pretty strongly wedded to the idea of TEX remaining compatible with its original behavior. At the same time, alterations to TEX (with a compatability switch) have greatly changed its functionality. I think the time has long since arrived when we should rethink some of the original design decisions that went into TEX almost 30 years ago.

The first relates to color; I discussed this issue earlier.

Another aspect has to do with the math infix operators (like `\over`). The infix operators are what cause the four math styles (`\displaystyle`, `\textstyle`, `\scriptstyle`, and `\scriptscriptstyle`) to be handled in such an awkward way. When a math list is being compiled, you do not know for sure which style you are in — all because of the existence of the infix operators. If these are done away with, the macro writer has much more effective control. And there is no compelling need for infix math operators: it was simply a decision, and it was one with (um) strong consequences for TEX.

Consider what happens when you enter the output routine: some items in the Main Vertical List are discarded, never to be recovered. Information should not be discarded; an effective alternative would be to present the items in `\box255`, but marked as discarded for the purposes of that tour of the output routine. If the output routine throws the contents of `\box255` back onto the MVL, the items marked discarded would reappear. The decision to discard items in this way is the source of enduring grief for those writing an output routine, as I did.

An annoying limitation of TEX is that there is a single class of `\mark` objects. An en-

hanced version of TEX with, say, 255 \marks, such as William Baxter's SuperTEX, enables one to take care of the output in an elegant way.

Yet another problem in TEX has to do with the algorithm that breaks paragraphs into lines: when that algorithm fails, it throws all the \badness into a single line. This means that upon failure, the output is very ugly. A better outcome would involve a softer landing.

Finally, the font data is static rather than dynamic: once read in, it can never be superseded or discarded. This misfeature stems from the notion that a font's data would be read in once and for all at the beginning of a job. Who would ever want to do otherwise?

Let us put these matters into perspective. TEX, in its day, had been considered a relatively large program, yet its executable (that is, not counting any storage) is a mere one-eighth of a megabyte of machine-language code (this is the case for older computers with compact instruction encoding; 64-bit computers may differ). Modern computers are many orders of magnitude speedier than the DEC System 10 that TEX82 originally ran on. Even in storage, Small TEX is no longer "large": it could fit on the *stack* of a modern personal computer.

Many of the design decisions of TEX were compelled by the limited storage and speed of 1980-era computers. Such limitations now perhaps apply to, say, running TEX on an iPhone — but certainly do not apply to running TEX on a Mac Pro, where one could conceivably call TEX as a function with all of its storage in a stack frame. Indeed, multiple instances of TEX could run simultaneously, each with its own storage.

If TEX were to be redone today, one could retain all of its good features while jettisoning its bad decisions, making it into a much better engine than before. At the same time, it would still have a compatibility switch that would make it pass the trip test.

Under the hood, it might use garbage collection and other modern programming paradigms. It might be integrated into the runtime library of your operating system, putting TEX's calculations at the service of any program running on your computer. It might take advantage of the multiple cores present in your computer. The possibilities are enormous, once you get your mind free of the mental blinders.

DW: The comprehensive list of officers and board members of TUG at `http://tug.org/board.html` shows that you have been a board member since 1997, were secretary for four years, and vice president for two years after that. Karl Berry also tells me that you and your wife "came through" for TUG by operating the TUG office in 1997. How did you come to be involved in TUG governance, why do you continue to participate, what was the situation that led to you providing the TUG office function (for those of us not familiar with TUG history), and what did you learn for yourself or TUG from the experience?

AO: In late 1986, I called the TUG office to renew my membership and had a conversation with Patricia Monohon, then the Executive Director. Patricia, cashing in on the considerable goodwill in our relationship, asked me if I would be willing to assist her in organizing the TUG 1997 conference to be held at the University of San Francisco.

Glossing over a great deal of *tsuris*, in the months between then and July 1997,

- I decided to run for TUG President, and then changed my mind and ran for Board instead,
- was appointed to the board early by then TUG president Michel Goossens,
- agreed to supervise Marian Goldeen (my wife) in temporarily running the TUG office when Patricia completed her term of employment as TUG Executive Director

in about May of 1997,

- took responsibility for the TUG 1997 conference in San Francisco,
- handed over the TUG office at the end of that conference to the newly installed TUG President Mimi Jett,
- agreed to work as TUG Secretary.

I continued to serve on the TUG Executive Committee, first as Secretary, later as Vice-President, until 2003.

I saw my role on the Board as responding to a need for transparency and responsibility to the TUG membership. We also had to figure out a new business model where we could afford to have an employee. This was at a time when *TUGboat* was chronically late in getting out, and TUG members were wondering what benefit they were receiving in exchange for their dues. As it happened, TUG decided to distribute the TEX Live software as a benefit of membership, which did much to bolster our fortunes.

My service to TUG reflects my sense that, as a self-employed person who benefitted from the existence of the TEX software and the large base of users of TEX, I had an obligation to do what I could to see to the health of TUG, the organization whose purpose was to benefit users of TEX.

Now that we have Robin Laakso as Executive Director and Karl Berry as TUG President, I think we are much better off than at any time since I began serving on the Board.

DW: Thank you, Arthur, for participating in this interview. I have greatly enjoyed hearing what you had to say.